A New Approach to Speeding Up Topic Modeling

Jia Zeng, *Member, IEEE*, Zhi-Qiang Liu and Xiao-Qin Cao

Abstract—Latent Dirichlet allocation (LDA) is a widely-used probabilistic topic modeling paradigm, and recently finds many applications in computer vision and computational biology. This paper proposes a fast and accurate algorithm, active belief propagation (ABP), for training LDA. Usually training LDA requires repeated scanning of the entire corpus and searching the complete topic space. Confronted with massive corpus with large number of topics, such a training iteration is often inefficient and time-consuming. To accelerate the training speed, ABP actively scans partial corpus and searches partial topic space for topic modeling, saving enormous training time in each iteration. To ensure accuracy, ABP selects only those documents and topics that contribute to the largest residuals within the residual belief propagation (RBP) framework. On four real-world corpora, ABP performs around 10 to 100 times faster than some of the major state-of-the-art algorithms for training LDA, while retains a comparable topic modeling accuracy.

Index Terms—Latent Dirichlet allocation, topic models, residual belief propagation, active belief propagation, message passing, factor graph, Markov random fields, Gibbs sampling, variational Bayes.

1 Introduction

The probabilistic topic modeling techniques divide the non-zero elements in the document-word matrix into several thematic groups called topics, which have found many real-world applications in text mining, computer vision, and computational biology [1]. As the simplest topic model, latent Dirichlet allocation (LDA) [2] represents each document as a mixture of topics and each topic as a multinomial distribution over the fixed vocabulary. From the labeling point of view, LDA assigns the hidden topic labels to explain the observed words in document-word matrix [3]. This labeling process defines a joint probability distribution over the hidden labels and the observed words. Employing the Bayes' rule, we can infer topic labels from observed words by computing the posterior distribution of the hidden variables given the observed variables from their joint probability. Such inference techniques have been widely-used for learning probabilistic graphical models within the Bayesian framework [4]. Recent approximate inference methods for LDA [5] fall broadly into three categories: variational Bayes (VB) [2], collapsed Gibbs sampling (GS) [6], and loopy belief propagation (BP) [3].

VB is a variational message passing method [7] that uses a set of factorized variational distributions to approximate the objective joint distribution by minimizing the Kullback-Leibler (KL) divergence between them. Because there is always a gap between the variational distribution and the true joint distribution, VB introduces bias when training LDA. In contrast, GS approximates this joint distribution by the Markov Chain Monte Carlo (MCMC) process, which is theoretically more accurate but converges practically much slower than VB. Representing LDA by a factor graph [4], [8], BP directly infers the marginal probability of hidden topics over observed words called *messages* from their joint probability, so that it is currently the most accurate algorithm for training LDA. Also, the message passing process of BP converges much faster than GS [3]. Similar BP implementation has also been discussed as the CVB0 algorithm within the mean-field framework [5], [9].

However, all above algorithms for training LDA require multiple iterations of scanning the entire corpus and searching the complete topic space. So, the computational cost increases linearly with the number of documents D, the number of topics K, and the number of training iterations T. Confronted with massive corpora with large number of topics, these conventional algorithms are often inefficient for fast topic modeling because of high per-iteration cost. For example, when $D = 10^6$, $K = 10^3$ and $T = 500$, GS consumes around forty hours to infer the hidden topics. Therefore, we are in great need of the fast topic modeling algorithms for real-world massive corpora.

In this paper, we propose a fast topic modeling algorithm for training LDA: active belief propagation (ABP). The basic idea is to actively select partial corpus and search partial topic space at each training iteration. In theory, ABP is based on the residual BP (RBP) [10] framework, which uses an informed scheduling for asynchronous message passing. First, we define the message residual as the absolute difference between messages at successive training iterations. The residuals evaluate the convergence speed of messages, where the larger residuals correspond to the faster convergence of messages. Second, through sorting residuals at each iteration, we actively select those documents and topics with the largest residuals for message updating and passing. For example, if ABP selects 10% documents and 10% topics for message passing at each iteration, it may be 100

- *J. Zeng is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China. To whom correspondence should be addressed. E-mail: j.zeng@ieee.org.*
- *Z.-Q. Liu and X.-Q. Cao are with the School of Creative Media, City University of Hong Kong, Hong Kong, China.*

times faster than conventional BP algorithms. Similar strategies that select to optimize those fast-converging messages have been discussed in active learning with statistical models [11]. Extensive experiments on several real-world corpora have demonstrated that ABP can achieve a significant speedup, while can retain a similar topic modeling accuracy as compared with current state-of-the-art algorithms for training LDA.

The proposed ABP algorithm can be also interpreted as a fast expectation-maximization (EM) algorithm [12]. At the E-step, ABP infers and updates partial messages with the largest residuals. At the M-step, ABP maximizes the likelihood by estimating parameters based on partial messages. After multiple iterations, it often converges at the local maximum of the joint probability. Because the residuals reflect the convergence speed of messages, ABP focuses all computational resources on inferring and updating those fast-converging messages at each iteration. In this way, it may achieve almost the same performance as BP. Therefore, ABP has the potential to be extended for learning other finite mixture models based on EM algorithms, such as Gaussian mixture models [4]. Similar principle can be also applied to developing fast k-means clustering algorithm [4].

In Section 2 we review some recent fast topic modeling techniques, and in Section 3 we introduce the factor graph interpretation of LDA and the residual BP (RBP) algorithm for training LDA. Section 4 proposes the active belief propagation (ABP) algorithm based on the residual belief propagation (RBP) framework. Section 5 compares ABP with the state-of-the-art algorithms on real-world text corpora. Finally, we draw conclusions and envision future work in Section 6.

2 Fast Topic Modeling Techniques

Currently, online and parallel learning algorithms are two major fast topic modeling techniques. Online learning algorithms for LDA [13], [14] partition the entire corpus into a set of mini-batches. For each mini-batch, online learning algorithms converge significantly faster than offline algorithms, and thus save enormous training time by reducing the total number of iterations T. For example, online GS [13] algorithms have two strategies. The first, called incremental LDA, periodically resamples the topic assignments for previously analyzed words. The second approach uses particle filtering instead of GS to assign the current topic based on previous topic configurations. Despite of faster speed, none of the strategies perform as well as offline GS algorithms. On the other hand, online VB algorithms [14] are based on online stochastic optimization, which can converge to a local optimum of the offline VB objective function. However, they require manually setting several free parameters including the mini-batch size, which controls the best topic modeling performance based on heuristics.

Parallel learning algorithms for LDA [15], [16] distribute large data sets across separate processors to accelerate the topic modeling process. For example, parallel GS algorithms [15] approximate the asynchronous GS sampling process by synchronous updating of the global topic distributions. Parallel VB algorithms [16] use the Map-Reduce technique to scale the probabilistic topic modeling. While these parallel techniques often provide almost the same topic modeling performance as single-processor algorithms, they require expensive parallel hardware so that the performance-to-price ratio remains unchanged. Moreover, parallel algorithms often consume additional resources to communicate and synchronize the topic modeling results from multiple processors. In practice, 1024 processors can achieve only around 700 times speedup corresponding to a low parallel efficiency of approximately 0.7 [15]. Finally, it is often difficult to write and debug parallel algorithms on distributed systems.

Current online and parallel techniques ignore handling the large number of topics K in massive corpora, which also increases the computational cost. The fast GS (FGS) algorithm [17] introduces the upper bound on the normalization factor using the Hölder's inequality. Without visiting all possible K topics, FGS is able to draw the target topic sample from the approximate posterior probability normalized by the upper bound. By refining the upper bound to approximate the true normalization factor, FGS samples topic label equal to that drawn from the true posterior probability as GS does. In this sense, FGS yields exactly the same result as GS but with much less computation.

3 Residual Belief Propagation

The probabilistic topic modeling task can be interpreted as a labeling problem, in which the objective is to assign a set of semantic topic labels, $\mathbf{z} = \{z^k_{w,d}\}$, to explain the observed document-word co-occurrence matrix, $\mathbf{x} = \{x_{w,d}\}$. The notations $1 \le w \le W$ and $1 \le d \le D$ are the word index in vocabulary and the document index in corpus. The notation $1 \le k \le K$ is the topic index. The nonzero element $x_{w,d} \neq 0$ denotes the number of word counts at the index $\{w, d\}$.

Fig. 1A is the three-layer graphical representation of LDA [2]. The document-specific topic proportion $\theta_d(k)$ generates a topic label, $z^k_{w,d,i} \in \{0, 1\}, \sum_{k=1}^K z^k_{w,d,i} = 1$, which in turn generates each observed word token n at the index $\{w, d\}$ based on the topic-specific multinomial distribution $\phi_k(w)$ over the vocabulary words. Both multinomial parameters $\theta_d(k)$ and $\phi_k(w)$ are generated by two Dirichlet distributions with hyperparameters α and β, respectively. For simplicity, we consider only a smoothed LDA with fixed symmetric Dirichlet hyperparameters α and β [6]. The plates indicate replications. For example, there are K topics, the document repeats D times in the corpus, the word repeats N_d times in the document d, and the vocabulary size is W.

Fig. 1B interprets Fig. 1A by the two-layer factor graph [4], [8]. The notation $z^k_{w,d} = \sum_{i=1}^{x_{w,d}} z^k_{w,d,n}/x_{w,d}$ is

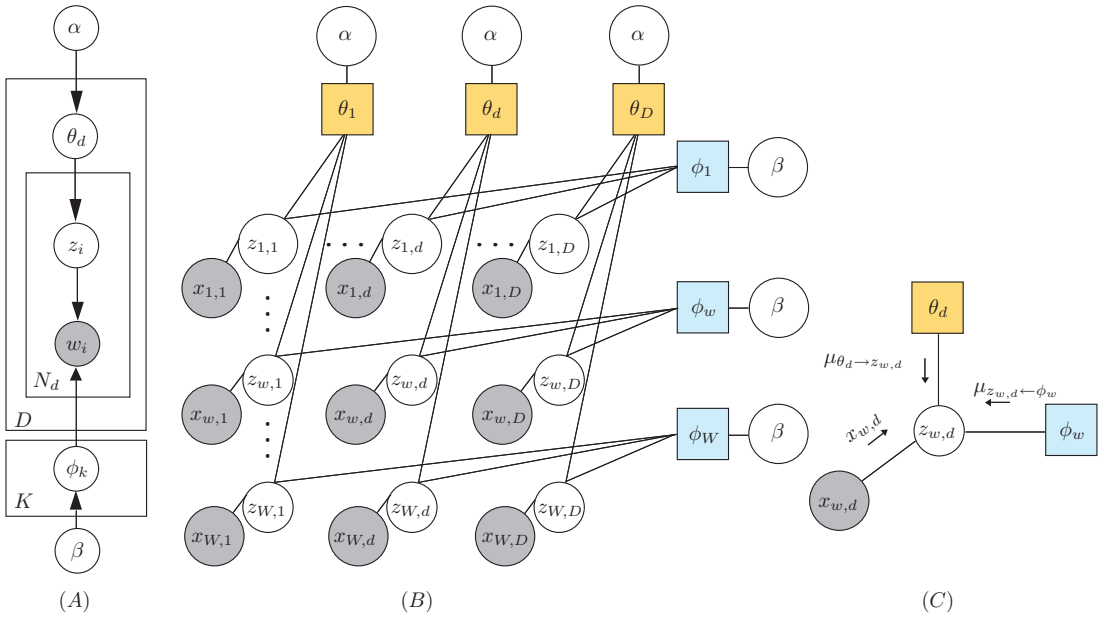

Fig. 1. (A) Three-layer graph, (B) two-layer factor graph, and (C) message passing for LDA.

the average topic labeling configuration over all word tokens $1 \leq i \leq x_{w,d}$ at the index $\{w, d\}$, where $x_{w,d}$ is the number of word counts at the index $\{w, d\}$. Note that $x_{w,d} z^k_{w,d} = \sum_{i=1}^{x_{w,d}} z^k_{w,d,i}$ recovers the original topic configuration over the word tokens in Fig. 1A. The factors θ_d and ϕ_w are denoted by squares, and their connecting variables $z_{w,d}$ are denoted by circles. The document-word co-occurrence matrix $\mathbf{x} = \{x_{w,d}\}$ is denoted by shaded circles. The factor θ_d connects topic labels $\mathbf{z}_{\cdot,d}$ on different word indices within the same document d, while the factor ϕ_w connects topic labels $\mathbf{z}_{w,\cdot}$ on the same word index w but in different documents. We absorb the observed word index w as the index of ϕ_w, which is similar to absorbing the observed document d as the index of θ_d in Fig. 1A. Notice that Fig. 1A and Fig. 1B have the same topology because both θ_d and ϕ_w have the same neighbors.

Integrating out parameters $\{\theta, \phi\}$ based on Dirichlet-Multinomial conjugacy yields the joint probability [18] of LDA in Fig. 1A,

$$P(\mathbf{x}, \mathbf{z}|\alpha, \beta) \propto \prod_{d=1}^{D} \prod_{k=1}^{K} \frac{\Gamma(\sum_{w=1}^{W} x_{w,d} z^k_{w,d} + \alpha)}{\Gamma[\sum_{k=1}^{K} (\sum_{w=1}^{W} x_{w,d} z^k_{w,d} + \alpha)]} \times \prod_{w=1}^{W} \prod_{k=1}^{K} \frac{\Gamma(\sum_{d=1}^{D} x_{w,d} z^k_{w,d} + \beta)}{\Gamma[\sum_{w=1}^{W} (\sum_{d=1}^{D} x_{w,d} z^k_{w,d} + \beta)]} = \prod_{d=1}^{D} f_{\theta_d}(\mathbf{x}_{\cdot,d}, \mathbf{z}_{\cdot,d}, \alpha) \prod_{w=1}^{W} f_{\phi_w}(\mathbf{x}_{w,\cdot}, \mathbf{z}_{w,\cdot}, \beta), \quad (1)$$

where $f(\cdot)$ is the factor function, and $\mathbf{z}_{\cdot,d} = \{z_{w,d}, \mathbf{z}_{-w,d}\}$ and $\mathbf{z}_{w,\cdot} = \{z_{w,d}, \mathbf{z}_{w,-d}\}$ denote subsets of the variables in Fig. 1B. Thus, the joint probability (1) of LDA can be re-written as the product of factor functions [4] of the factor graph in Fig. 1B. Because Fig. 1A and B encode the same joint probability, they represent the same LDA model but from different perspectives.

Loopy BP [3] is an approximate inference method for factor graphs with loops in Fig. 1B. Rather than directly computing the posterior probability $p(\mathbf{z}|\mathbf{x})$, we calculate the marginal probability $p(z^k_{w,d}, x_{w,d}|\mathbf{z}^k_{-w,-d}, \mathbf{x}_{-w,-d})$, referred to as *message*, $\mu(z^k_{w,d} = 1) = \mu_{w,d}(k)$, which can be normalized efficiently using a local computation, i.e., $\sum_{k=1}^{K} \mu_{w,d}(k) = 1, 0 \leq \mu_{w,d}(k) \leq 1$. According to the Markov property and Fig. 1B, we obtain

$$p(z^k_{w,d}, x_{w,d}|\mathbf{z}^k_{-w,-d}, \mathbf{x}_{-w,-d}) \propto p(z^k_{w,d}, x_{w,d}|\mathbf{z}^k_{-w,d}, \mathbf{x}_{-w,d}) p(z^k_{w,d}, x_{w,d}|\mathbf{z}^k_{w,-d}, \mathbf{x}_{w,-d}), \quad (2)$$

where $-w$ and $-d$ denote all word indices except w and all document indices except d, and the notations $\mathbf{z}_{-w,d}$ and $\mathbf{z}_{w,-d}$ represent all possible neighboring labeling configurations. From the message passing view, $p(z^k_{w,d}, x_{w,d}|\mathbf{z}^k_{-w,d}, \mathbf{x}_{-w,d})$ is the neighboring message $\mu_{\theta_d \to z_{w,d}}(k)$ sent from the factor node θ_d, and $p(z^k_{w,d}|\mathbf{z}^k_{w,-d}, \mathbf{x}_{w,-d})$ is the other neighboring message $\mu_{\phi_w \to z_{w,d}}(k)$ sent from the factor node ϕ_w. Eq. (2) encourages only K smooth topic configurations within the neighborhood system.

Using the Bayes' rule and the joint probability (1), we expand (2) as

$$\mu_{w,d}(k) \propto \frac{p(\mathbf{z}^k_{\cdot,d}, \mathbf{x}_{\cdot,d})}{p(\mathbf{z}^k_{-w,d}, \mathbf{x}_{-w,d})} \times \frac{p(\mathbf{z}^k_{w,\cdot}, \mathbf{x}_{w,\cdot})}{p(\mathbf{z}^k_{w,-d}, \mathbf{x}_{w,-d})}, \propto \frac{\sum_{-w} x_{-w,d} z^k_{-w,d} + \alpha}{\sum_{k=1}^{K} (\sum_{-w} x_{-w,d} z^k_{-w,d} + \alpha)} \times \frac{\sum_{-d} x_{w,-d} z^k_{w,-d} + \beta}{\sum_{w=1}^{W} (\sum_{-d} x_{w,-d} z^k_{w,-d} + \beta)}, \quad (3)$$

where we use the property, $\Gamma(x+1) = x\Gamma(x)$, to cancel the common terms in both nominator and denominator.

Eq. (3) updates the message of $z_{w,d}^k$ if its neighboring topic configuration $\{\mathbf{z}_{-w,d}^k, \mathbf{z}_{w,-d}^k\}$ is known. However, due to uncertainty, we know only the neighboring messages rather than the precise topic configuration. So, we replace the topic configuration by the messages in Eq. (3) and obtain the message update equation,

$$\mu_{w,d}(k) \propto \frac{\boldsymbol{\mu}_{-w,d}(k)+\alpha}{\sum_k[\boldsymbol{\mu}_{-w,d}(k)+\alpha]} \times \frac{\boldsymbol{\mu}_{w,-d}(k)+\beta}{\sum_w[\boldsymbol{\mu}_{w,-d}(k)+\beta]}, \quad (4)$$

where

$$\boldsymbol{\mu}_{-w,d}(k) = \sum_{-w} x_{-w,d}\mu_{-w,d}(k), \quad (5)$$

$$\boldsymbol{\mu}_{w,-d}(k) = \sum_{-d} x_{w,-d}\mu_{w,-d}(k). \quad (6)$$

After updating, we need to normalize messages locally by the normalization factor Z,

$$Z = \sum_{k=1}^{K} \mu_{w,d}(k), \quad (7)$$

which requires K iterative computation.

Messages are passed from variables to factors, and in turn from factors to variables until convergence or the maximum number of iterations. Notice that we need pass only messages for which $x_{w,d} \neq 0$. Because $\mathbf{x}$ is a very sparse matrix, Eq. (4) is computationally fast by sweeping only nonzero elements in the sparse matrix. Based on the inferred messages, we estimate the multinomial parameters θ and ϕ by the expectation-maximization (EM) algorithm [18] as follows,

$$\theta_d(k) = \frac{\boldsymbol{\mu}_{\cdot,d}(k)+\alpha}{\sum_k[\boldsymbol{\mu}_{\cdot,d}(k)+\alpha]}, \quad (8)$$

$$\phi_w(k) = \frac{\boldsymbol{\mu}_{w,\cdot}(k)+\beta}{\sum_w[\boldsymbol{\mu}_{w,\cdot}(k)+\beta]}. \quad (9)$$

The synchronous BP [3] updates all messages simultaneously at iteration t based on the messages at previous iteration $t-1$. Although in practice this schedule often converges, it uses more number of training iterations until convergence than VB. Notice that the fast convergence is a desired property for online learning [14] or distributed learning [16] for LDA. Therefore, we adopt RBP [10] for the rapid convergence based on the informed schedule of asynchronous message passing.

The asynchronous schedule updates the message of each variable in a certain order, which is in turn used to update other neighboring messages immediately at each iteration t. The basic idea of RBP is to select the best update order based on the messages' residuals $r_{w,d}(k)$, which are defined as the p-norm of difference between two message vectors at successive iterations. Here, we choose the L_1 norm with $p=1$,

$$r_{w,d}(k) = x_{w,d}|\mu_{w,d}^t(k) - \mu_{w,d}^{t-1}(k)|, \quad (10)$$

where $x_{w,d}$ is the number of word counts. If we sequentially update messages in a descending order of $r_{w,d} = \sum_k r_{w,d}(k)$ at each iteration, the RBP algorithm theoretically converges faster or more often to a fixed point than synchronous BP [10]. Because RBP always updates messages with top largest residuals first, it efficiently accelerates the convergence than synchronous BP. Moreover, the updated messages with top largest residuals will in turn influence their neighboring messages quickly, which accounts for the fast convergence property of RBP. More details on theoretical and experimental analysis of RBP's convergence property can be found in [10].

4 Active Belief Propagation

In this section, we propose ABP within the RBP framework. We describe in detail how to actively select partial documents and topics at each iteration for fast topic modeling. We also provide an alternative implementation of ABP. Finally, we discuss ABP's relationship to previous algorithms.

4.1 The ABP Algorithm

At each training iteration, ABP actively selects partial documents $\lambda_d D$ from corpus for message updating and passing, where the parameter $\lambda_d \in (0,1]$ denotes the scanned corpus proportion. For each document, ABP searches partial topic space $\lambda_k K$ for message updating and passing, where the parameter $\lambda_k \in (0,1]$ denotes the proportion of the searched topic space. In theory, ABP costs only $\lambda_d\lambda_k$ training time of BP at each iteration.

Based on (10), we define the residuals at topics for specific documents as

$$r_d(k) = \sum_w r_{w,d}(k), \quad (11)$$

and accumulate all topic residuals at documents as

$$r_d = \sum_k r_d(k). \quad (12)$$

After each iteration, we sort $r_d(k)$ in a descending order for each document, and select the subset topics $\lambda_k K$ with the largest residuals $r_d(k)$. We also sort r_d in a descending order for all documents, and select the subset documents $\lambda_d D$ with the largest residuals r_d. In the following iteration, we update and pass only messages for the subset documents $\lambda_d D$ and the subset topics $\lambda_k K$, and thus save enormous training time in each loop.

According to the selected $\lambda_k K$ topics, we need to normalize local messages. At the first iteration $t=1$, ABP performs the same as the conventional BP, which updates and normalizes all messages for all topics. In the later iterations, based on residuals, ABP actively selects the subset $\lambda_k K$ topics for message updating and passing. We normalize local messages for the selected topics $k \in \lambda_k K$ by

$$\hat{\mu}_{w,d}^t(k) = \frac{\mu_{w,d}^t(k)}{\sum_k \mu_{w,d}^t(k)} \times \sum_k \hat{\mu}_{w,d}^{t-1}(k), k \in \lambda_k K, \quad (13)$$

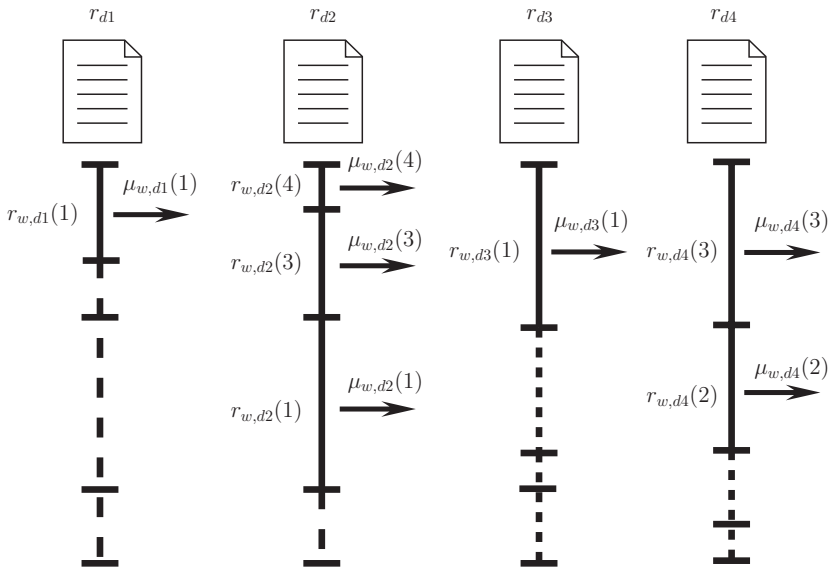

Fig. 2. Active message passing based on the sorted residuals.

where $\hat{\mu}_{w,d}^{t-1}$ is the normalized message in the previous iteration, $\hat{\mu}_{w,d}^{t}(k)$ is the normalized message in the current iteration, and $\mu_{w,d}^{t}(k)$ is the unnormalized message updated according to (4). In this way, we require only $\lambda_k K$ iterations for the local message normalization to avoid calculating the normalization factor Z in (7) with K computation.

At a negligible computational cost, the sorted residuals can be refined during message passing process. The computational cost of initial sorting (11) and (12) using the standard "quick sort" algorithm are $\mathcal{O}(K \log K)$ and $\mathcal{O}(D \log D)$, respectively. If the successive residuals are in almost sorted order, only a few swaps will restore the sorted order by the standard "insertion sort" algorithm, thereby saving lots of sorting time.

Fig. 2 shows an example of active message passing. The solid and dashed segments denote normalized messages in proportion. The solid segments and arrows denote those selected messages for updating and passing, while the dashed segments denote those unchanged messages. Suppose that documents $d1$, $d2$, $d3$ and $d4$ have top largest residuals r_{d1}, r_{d2}, r_{d3} and r_{d4}. We select these four documents in the subset $\lambda_d D$ for active message passing. For the document $d1$, only $r_{w,d1}(1)$ is in the subset $\lambda_k K$ so that the message $\mu_{w,d1}(1)$ is updated and passed. For the document $d2$, three residuals $r_{w,d2}(4)$, $r_{w,d2}(3)$ and $r_{w,d2}(1)$ are ranked in the subset $\lambda_k K$, and thus the messages $\mu_{w,d2}(4)$, $\mu_{w,d2}(3)$ and $\mu_{w,d2}(1)$ are updated and passed. Similar rules are applied to documents $d3$ and $d4$. Because only partial messages are updated and passed, ABP consumes significantly less computation than BP in each loop.

The residuals are dynamically refined and sorted at each iteration. Due to convergence, the top residuals in the previous iterations may be ranked lower in the following iterations. As a result, those unchanged messages still have the chance to be updated and passed. For example in Fig. 2, those dashed segments still have the chance to become solid segments in later iterations. In this sense, ABP remains the same message information as BP. Given the fixed training iterations T, ABP scans specific partial documents and searches specific partial topics with more iterations than others. In contrast, the BP algorithm scans all documents and searches all topics with equal number of iterations.

Fig. 3 summarizes the ABP algorithm. When $t = 1$, ABP scans the entire corpus and searches the complete topic space, and in the meanwhile computes and sorts residuals by the "quick sort" algorithm. For $2 \leq t \leq T$, ABP actively selects the subset documents $\lambda_d D$ and the subset topics $\lambda_k K$ for message updating and passing. After each iteration, ABP dynamically refines and sorts residuals by the "insertion sort" algorithm. The ABP algorithm terminates when T is reached or convergence condition is satisfied. In practice, ABP costs more than $\lambda_d \lambda_k$ training time of BP at each iteration due to additional time for refining and sorting residuals.

4.2 An Alternative Implementation for ABP

For massive corpora, initially sorting r_d in (12) requires at most a computational complexity of $\mathcal{O}(D \log D)$. This cost is very high when D is very large, for example, $D = 10^8$. Alternatively, we may define residuals at the fixed vocabulary,

$$r_w(k) = \sum_d r_{w,d}(k), \tag{14}$$

and

$$r_w = \sum_k r_w(k). \tag{15}$$

We may sort r_w with a significantly less computational cost of $\mathcal{O}(W \log W)$ because W is often a fixed value

input : $\mathbf{x}, K, T, \alpha, \beta, \lambda_d, \lambda_k$.
output : θ_d, ϕ_w.
$\mu^0_{w,d}(k) \leftarrow$ random initialization and normalization;
for $d \leftarrow 1$ **to** D **do**
  for $k \leftarrow 1$ **to** K **do**
    $\mu^1_{w,d}(k) \propto \frac{\boldsymbol{\mu}^0_{-w,d}(k)+\alpha}{\sum_k[\boldsymbol{\mu}^0_{-w,d}(k)+\alpha]} \times \frac{\boldsymbol{\mu}^0_{w,-d}(k)+\beta}{\sum_w[\boldsymbol{\mu}^0_{w,-d}(k)+\beta]}$;
    $\mu^1_{w,d}(k) \leftarrow$ normalize$(\mu^1_{w,d}(k))$;
    $r^1_{w,d}(k) \leftarrow x_{w,d}|\mu^1_{w,d}(k) - \mu^0_{w,d}(k)|$;
  end
  $\lambda_k K \leftarrow$ quick sort$(r^t_d(k)$, 'descend');
end
$\lambda_d D \leftarrow$ quick sort$(r^t_d$, 'descend');
for $t \leftarrow 2$ **to** T **do**
  for $d \in \lambda_d D$ **do**
    for $k \in \lambda_k K$ **do**
      $\mu^t_{w,d}(k) \propto \frac{\boldsymbol{\mu}^{t-1}_{-w,d}(k)+\alpha}{\sum_k[\boldsymbol{\mu}^{t-1}_{-w,d}(k)+\alpha]} \times \frac{\boldsymbol{\mu}^{t-1}_{w,-d}(k)+\beta}{\sum_w[\boldsymbol{\mu}^{t-1}_{w,-d}(k)+\beta]}$;
      $\mu^t_{w,d}(k) \leftarrow$ normalize$(\mu^t_{w,d}(k))$;
      $r^t_{w,d}(k) \leftarrow x_{w,d}|\mu^t_{w,d}(k) - \mu^{t-1}_{w,d}(k)|$;
    end
    $\lambda_k K \leftarrow$ insertion sort$(r^t_d(k)$, 'descend');
  end
  $\lambda_d D \leftarrow$ insertion sort$(r^t_d$, 'descend');
end
$\theta_d(k) \leftarrow [\boldsymbol{\mu}^T_{\cdot,d}(k)+\alpha]/\sum_k[\boldsymbol{\mu}^T_{\cdot,d}(k)+\alpha]$;
$\phi_w(k) \leftarrow [\boldsymbol{\mu}^T_{w,\cdot}(k)+\beta]/\sum_w[\boldsymbol{\mu}^T_{w,\cdot}(k)+\beta]$;

Fig. 3. The ABP algorithm.

when D is a huge number for a massive copora. In this case, ABP actively selects partial vocabulary words, $\lambda_w W, \lambda_w \in (0, 1]$, for topic modeling at each iteration. Due to page limitation of this paper, we do not show the experimental results of this alternative. Interested readers may find source codes of this implementation in [19].

4.3 Relationship to Previous Algorithms

The proposed ABP algorithm adopts a simple idea to accelerate the topic modeling process based on residuals. At each iteration, it selects only a mini-batch from the entire corpus for message updating and passing. From this perspective, ABP resembles the online learning algorithms [13], [14] but with two main distinctions. First, online algorithms randomly read each mini-batch from a data stream without selection. Second, online algorithms use each mini-batch only once, which may cause a lower topic modeling accuracy [13]. Technically, ABP can converge to the local maximum of the objective function as BP, while online algorithms require specific parameter settings to achieve this goal [14].

Unlike the parallel algorithms [15], [16] for LDA, ABP uses a single-processor to perform topic modeling without complicated parallel architectures. However, due to memory limitation, it is hard to fit massive corpora with large number of topics into the memory of a common PC (for example, 3G RAM). We have two straightforward strategies to solve this problem. First, we may extend ABP to online and parallel learning. Online learning requires a fixed memory for each mini-batch data, while parallel learning has the large memory space based on parallel architectures. Second, we may adopt the block optimization framework [20], which sequentially reads blocks of data from the hard disk into memory.

The FGS algorithm [17] is an important improvement over the conventional GS algorithm when K is very large. The basic idea is to combine both message update and sampling process together by introducing an upper bound $\hat{Z}$ for the normalization factor Z. FGS further refines the upper bound to approximate the true normalization factor, so that the sampled topic label equals that drawn from the true posterior probability. Similar to FGS, we may define an upper bound and pass only top largest segments $\mu_{w,d}(k)/\hat{Z}$ in the BP algorithm to reduce the computation. However, this strategy does not work because we will not be able to update and pass those short segments in Fig. 2, leading to serious loss of information. While such a strategy is faster, its accuracy is much lower than the conventional BP algorithm. To avoid information loss, ABP is based on the RBP framework by dynamically refining and sorting residuals, which determines the subsect $\lambda_k K$ topic space at each iteration. Through the dynamical residual sorting, ABP has the chance to scan all documents and search topic space without information loss.

TABLE 1
Statistics of four document data sets.

Data sets	D	W	N_d	W_d
NIPS	1500	12419	311.8	217.6
ENRON	39861	28102	160.9	93.1
NYTIMES	15000	84258	328.7	230.2
PUBMED	80000	76878	68.4	46.7

5 Experiments

Our experiments aim to verify the accelerating effects of ABP compared with VB [2], GS [6], FGS [17], and BP [3] algorithms. We use four publicly available document data sets [17]: NIPS, ENRON, NYTIMES and PUBMED. Previous studies [17] revealed that the speedup effect is relatively insensitive to the total number of documents in the corpus. Because of the memory limitation, we randomly select 15000 documents from the original NYTIMES data set, and 80000 documents from the original PUBMED data set for experiments. Table 1 summarizes the statistics of four data sets, where D is the total number of documents in the corpus, W is the number of words in the vocabulary, N_d is the average number of word tokens per document, and W_d is the average number of word indices per document.

We randomly partition each data set into halves with one for training set and the other for test set. We calculate

the training perplexity [5] on the training set after 500 iterations as follows,

$$\mathcal{P} = \exp\left\{ -\frac{\sum_{w,d} x_{w,d} \log\left[\sum_k \theta_d(k)\phi_w(k)\right]}{\sum_{w,d} x_{w,d}} \right\}. \quad (16)$$

Usually, the training perplexity will decrease with the increase of number of training iterations. The algorithm often converges if the change of training perplexity at successive iterations is less than a predefined threshold. In our experiments, we set the threshold as one because the decrease of training perplexity is very small after satisfying this threshold.

The predictive perplexity for the unseen test set is computed as follows [5]. On the training set, we estimate ϕ from the same random initialization after 500 iterations. For the test set, we randomly partition each document into 80% and 20% subsets. Fixing ϕ, we estimate θ on the 80% subset by training algorithms from the same random initialization after 500 iterations, and then calculate the predictive perplexity on the rest 20% subset,

$$\mathcal{P} = \exp\left\{ -\frac{\sum_{w,d} x^{20\%}_{w,d} \log\left[\sum_k \theta_d(k)\phi_w(k)\right]}{\sum_{w,d} x^{20\%}_{w,d}} \right\}, \quad (17)$$

where $x^{20\%}_{w,d}$ denotes word counts in the the 20% subset. The lower predictive perplexity represents a better generalization ability.

For all data sets, we fix the same hyperparameters as $\alpha = 2/K, \beta = 0.01$ [17]. The CPU time per iteration is measured after sweeping the entire data set. We report the average CPU time per iteration after $T = 500$ iterations, which practically ensures that GS and FGS converges in terms of training perplexity. For fair comparison, we use the same random initialization to examine all algorithms with 500 iterations. We implement all algorithms based on the MEX C++/MATLAB/Octave platform, where GS and FGS C++ source codes are the same as FastLDA [17]. To repeat our experiments, we have made all source codes and data sets publicly available [19]. All experiments are run on the Sun Fire X4270 M2 server with two 6-core 3.46 GHz CPUs and 128 GB RAMs.

5.1 Parameters λ_d and λ_k

First, we examine two parameters λ_d and λ_k in ABP on the relatively smaller NIPS data set. We are wondering how these two parameters would influence the topic modeling accuracy of ABP. The parameter $\lambda_d \in (0, 1]$ controls the proportion of documents to be scanned, and the parameter $\lambda_k \in (0, 1]$ defines the proportion of topics to be searched at each iteration. The smaller values correspond to the faster speed of ABP. We choose the training perplexity of ABP with 500 iterations when $\lambda_d = \lambda_k = 1$ as the benchmark. The relative training perplexity is the difference between this benchmark and the ABP's training perplexity with 500 iterations at other parameter values.

Fixing $\lambda_k = 1$, we change λ_d from 0.1 to 0.5 with the step 0.1. Fig. 4 shows the relative training perplexity as a function of λ_d when $K = \{100, 300, 500, 700, 900\}$. We see that the relative perplexity decreases when λ_d increases, which confirms the fact that scanning more documents at each iteration will yield the lower training perplexity under different topics K. Notice that the relative perplexity is larger when K is smaller, while it is smaller when K is larger. This phenomenon shows that when K is very large, even scanning a small portion of documents is enough to provide a comparable topic modeling accuracy. When $\lambda_d \geq 0.2$ we also see that the relative perplexity is less than 20, which in practice is a negligible difference on a common document data set. Therefore, we think that $\lambda_d = 0.2$ is a safe bound to guarantee a good topic modeling accuracy with a relatively faster speed. Although smaller $\lambda_d < 0.2$ will produce faster speed, it will cause an obvious degrade of topic modeling accuracy.

Fixing $\lambda_d = 1$, we also change λ_k from 0.1 to 0.5 with the step 0.1. Fig. 5 shows the relative training perplexity as a function of λ_k when $K \in \{100, 300, 500, 700, 900\}$. Surprisingly, there is no big difference when $\lambda_k = 0.1$ and $\lambda_k = 0.5$ especially when $K = 700$. This phenomenon implies that only a small proportion of topics plays the major role when K is very large. When $\lambda_k \leq 0.5$, ABP achieves even a lower perplexity value than ABP with $\lambda_k = 1$. The reason is that most documents have very sparse topic proportions when K is very large, and thus searching partial topic space is enough to yield a comparable topic modeling accuracy. We wonder whether λ_k can be even smaller when K is very large, e.g., $K \in \{1500, 2000\}$. On NIPS data set, ABP with $\lambda_k = 0.05$ achieves 555.89 and 542.70 training perplexity, respectively. In contrast, ABP with $\lambda_k = 1$ achieves 543.90 and 533.97 training perplexity, respectively. The relative training perplexity is less than 2%. Therefore, we speculate that when K is very large like $K \geq 2000$, $\lambda_k K$ may be a constant, e.g., $\lambda_k K = 100$. In this case, the training time of ABP is independent of K. This bound $\lambda_k K = 100$ is reasonable because usually a common word is unlikely to be associated with more than 100 topics.

Users may set different parameters λ_d and λ_k for different speedup effects. To pursue the maximum speedup, we choose $\lambda_d = \lambda_k = 0.1$ referred to as ABP1 in the rest of our experiments. To ensure the topic modeling accuracy, we also choose the safe bound $\lambda_d = \lambda_k = 0.2$ referred to as ABP2. Obviously, ABP1 is much faster than ABP2 but with relatively lower accuracy in terms of training perplexity.

5.2 Predictive Perplexity

The predictive perplexity is a widely-used performance measure for the generalization ability [2], especially for different training algorithms of LDA. Fig. 6 compares the predictive perplexity as a function of topics $K =$

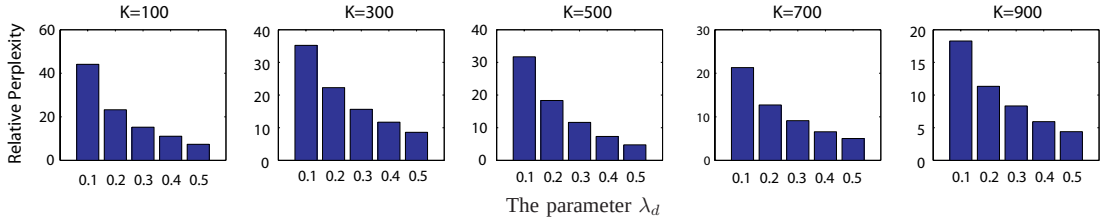

Fig. 4. Relative training perplexity as a function of the parameter λ_d when $K = \{100, 300, 500, 700, 900\}$.

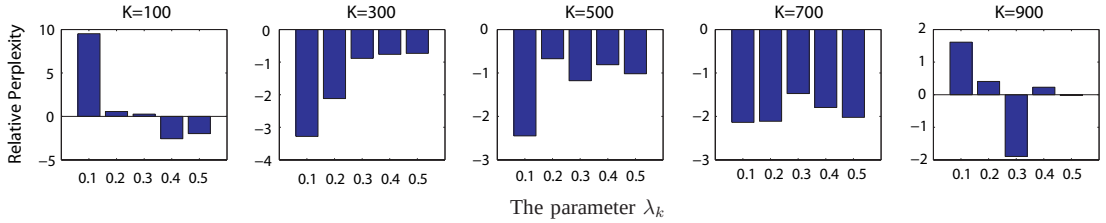

Fig. 5. Relative training perplexity as a function of the parameter λ_k when $K = \{100, 300, 500, 700, 900\}$.

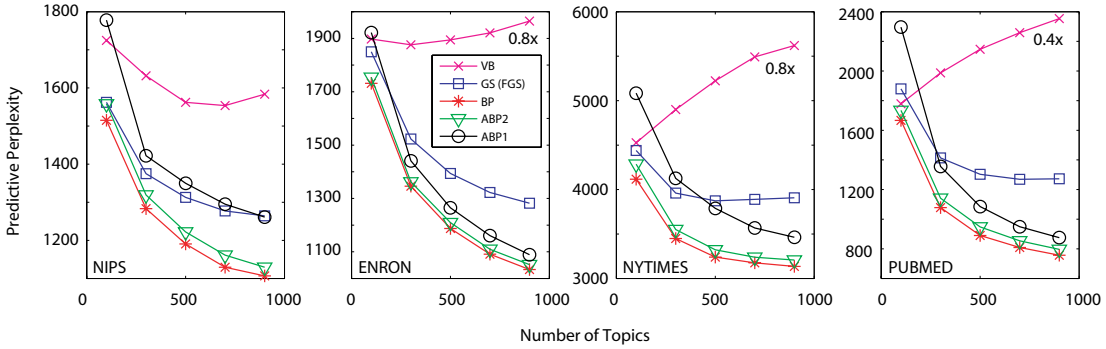

Fig. 6. Predictive perplexity as a function of topics $K = \{100, 300, 500, 700, 900\}$ on NIPS, ENRON, NYTIMES and PUBMED data sets. The notation $0.8x$ denotes the predictive perplexity is multiplied by 0.8.

$\{100, 300, 500, 700, 900\}$ by VB, GS/FGS, BP, ABP1 and ABP2 algorithms. FGS and GS have exactly the same topic modeling accuracy, so that their predictive perplexity curves overlap denoted by blue squares.

On ENRON, NYTIMES and PUBMED data sets, VB shows an obvious overfitting phenomenon, where the predictive perplexity increases with the number of topics K. Also, VB often yields the highest predictive perplexity. For a better visualization, we multiply VB's predictive perplexity value by 0.8 and 0.4 on ENRON, NYTIMES and PUBMED data sets, respectively. Therefore, under a large number of topics on large data sets, VB often generalizes badly to predict unseen test set. The major reason is that VB aims to optimize an approximate variational distribution with a gap from the objective joint distribution. Experiments show that this gap or bias is obvious in case of the massive data set with the large number of topics. Although VB may choose the proper hyperparameters to correct this bias [5], it is still unreliable for inferring thousands of topics from massive data sets.

On all data sets, BP consistently performs the best with the lowest predictive perplexity. Unlike VB, BP decreases the predictive perplexity with the increase of the number of topics K without overfitting. Theoretically, there is no gap because BP directly infers the marginal posterior probability called message from the objective joint probability. So, BP always yields the lowest predictive perplexity with the best generalization ability in practice.

With the parameters $\lambda_d = \lambda_k = 0.2$, ABP2 performs almost the same as BP, especially on ENRON and PUBMED data sets. This result shows that scanning 20% documents and searching 20% topics at each iteration is enough to provide a comparable topic modeling accuracy as BP. With the parameters $\lambda_d = \lambda_k = 0.1$, ABP1 provides a much higher predictive perplexity than BP. On the relatively smaller data set NIPS, ABP1 performs even worse than VB when $K = 100$. However, on the relatively large data set PUBMED and when K is large, e.g., $K = 900$, ABP1 shows a comparable predictive perplexity as BP. When compared with GS/FGS, ABP1 is worse under different topics on the relatively small NIPS set, but is much better on the large data sets such as ENRON, NYTIMES and PUBMED when K is large.

In conclusion, ABP2 often yields a comparable performance as BP, and outperforms GS/FGS and VB on all

data sets. While ABP1 has a higher predictive perplexity than ABP2 and BP on all data sets, it often outperforms GS/FGS and VB for larger data sets with larger number of topics.

5.3 Speedup Effects

Fig. 7 shows the CUP time (seconds) per iteration as a function of K for VB, GS, FGS, BP, ABP1 and ABP2. The training time of all algorithms increases linearly with K except FGS on the PUBMED data set. When K is very large, FGS [17] will locate the targeted topics more efficiently so that its training time does not increase linearly with K. Because FGS does not visit all possible topic space to sample the targeted topic, it is much faster than GS when K is very large. Notice that FGS still needs to scan the entire corpus to sample topics for each word token. VB consumes the longest training time because of its complicated digamma function calculation [3]. For a better illustration, we multiply VB's training time by 0.2. This result confirms that VB is unsuitable for fast topic modeling of massive data sets.

The training time of BP is close to that of GS when K is small, but is longer than that of GS when K is large on ENRON, NYTIMES and PUBMED data sets. The major reason is that BP requires visiting all topic space, while GS stops visiting the rest topic space until sampling the target topic. When $\lambda_d = \lambda_k = 0.1$, ABP1 in theory requires only $1/100$ training time of BP. However, ABP1 needs to sort and update residuals so that it on average consumes around $1/30$ training time of BP. When $\lambda_d = \lambda_k = 0.2$, ABP2 theoretically requires four times longer training time than ABP1. But in practice ABP1 runs around three times faster than ABP2. The speedup effects are significant. In conclusion, Fig. 7 demonstrates that both ABP1 and ABP2 are at present the fastest algorithms for training LDA.

Fig. 8 shows the ratio of CPU time per iteration over ABP1 as the benchmark. For a better illustration, we multiply the ratio of VB by 0.2. We see that ABP1 runs at least 100 times faster than VB on NIPS, NYTIMES and PUBMED data sets. On ENRON set, we also see that ABP1 is close to 100 times faster than VB. Such a significant speedup facilitates fast topic modeling of massive data sets. For example, if VB uses three months for training LDA, ABP1 consumes only one day for the same task with much better generalization ability shown in Fig. 6. ABP1 also runs around 20 to 40 times faster than both GS and BP. When compared with relatively faster FGS, ABP1 still runs around 10 to 20 times faster with comparable predictive perplexity. ABP2 consistently consumes 2 to 3 times more training time than ABP1. Both Figs. 6 and 8 reconfirm that the proposed ABP algorithm can be around 10 to 100 times faster than the current state-of-the-art algorithms, while retains a comparable accuracy in topic modeling of massive data sets. Therefore, we advocate ABP1 for fast topic modeling of the massive data set when D and K are large.

5.4 Convergence

Fig. 9 shows the training perplexity as a function of number of training iterations when $K = 500$. We see that GS and FGS converge at the same speed (overlapped curves) but converge much slower than both ABP2 and BP. Usually, GS and FGS uses $400 \sim 500$ iterations to achieve convergence. Among all algorithms, VB converges fastest, which often uses around $80 \sim 150$ iterations for convergence. Although VB sometimes achieves the lowest training perplexity on NIPS and NYTIMES data sets, it yields the highest predictive perplexity on unseen test set by overfitting. We see that ABP2 converges slightly slower than BP, which is caused by scanning partial corpus and searching partial topic space at each iteration. In our experiments, BP uses around $100 \sim 200$ iterations, while ABP2 uses around $300 \sim 500$ iterations to achieve convergence. On NYTIMES and PUBMED data sets, ABP2 converges with almost the same speed as GS and FGS algorithms. Furthermore, ABP1 converges significantly slower than ABP2 on all data sets. It often uses $500 \sim 800$ iterations for convergence. While at each iteration ABP1 scans 10% corpus and searches 10% topics for fast speed, it significantly slows down the convergence speed than BP. However, even if we compare the training time until convergence, ABP1 is still the fastest algorithm. Fig. 10 summarizes the training time ratio until convergence over ABP1. We see that ABP1 still runs around $20 \sim 40$ times faster than VB, $10 \sim 20$ times faster than GS, and $5 \sim 10$ times faster than both FGS and BP.

5.5 Topic Visualization

Fig. 11 shows the top ten words of ten topics extracted by VB (red), GS/FGS (blue), BP (black) and ABP (green) algorithms on the NIPS training set. For ABP, we choose $\lambda_d = \lambda_k = 0.5$ because $K = 10$ is small. We see that all algorithms can infer semantically meaningful topics, where most top ten words are consistent except the slightly different word ranking. There are two subjective measures for the interpretability of discovered topics: word intrusion and topic intrusion [21]. The former is the number of word intruders in each topic, while the latter is the number of topic intruders in each document. According to our experience, the word and topic intrusions are comparable among all algorithms. Given the similar interpretable topics, we advocate ABP for fast topic modeling of massive data sets.

6 Conclusions

Fast training algorithms for LDA has attracted intensive interests recently. For fast topic modeling the real-world massive corpora with large number of topics, the online and parallel learning algorithms are two straightforward strategies. The proposed ABP algorithm provides the third alternative for fast topic modeling, which may also inspire more efficient online and parallel algorithms. Extensive experiments on large data sets demonstrate that

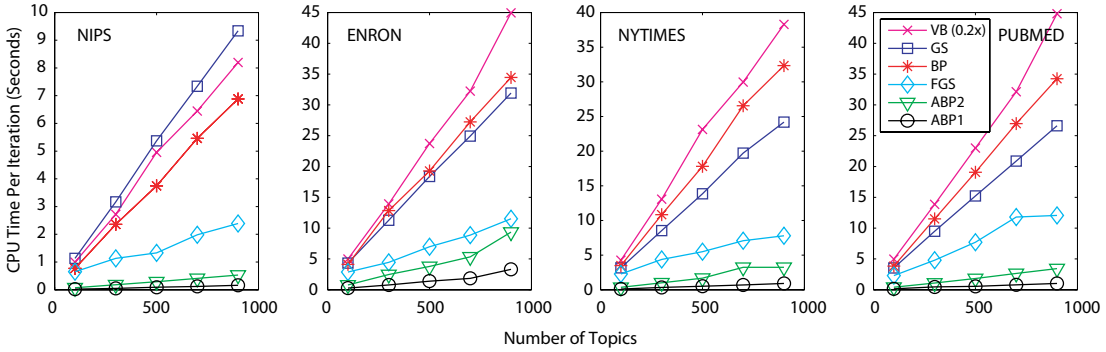

Fig. 7. CPU time per iteration (seconds) as a function of topics $K = \{100, 300, 500, 700, 900\}$ on NIPS, ENRON, NYTIMES and PUBMED data sets. The notation $0.2x$ denotes the training time is multiplied by 0.2.

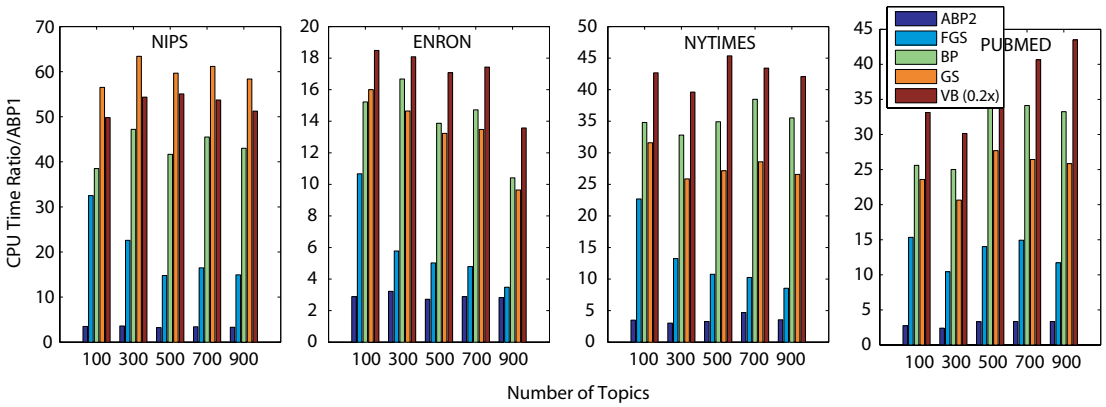

Fig. 8. Ratio of CPU time per iteration over ABP1 as a function of topics $K = \{100, 300, 500, 700, 900\}$ on NIPS, ENRON, NYTIMES and PUBMED data sets. The notation $0.2x$ denotes the predictive perplexity is multiplied by 0.2.

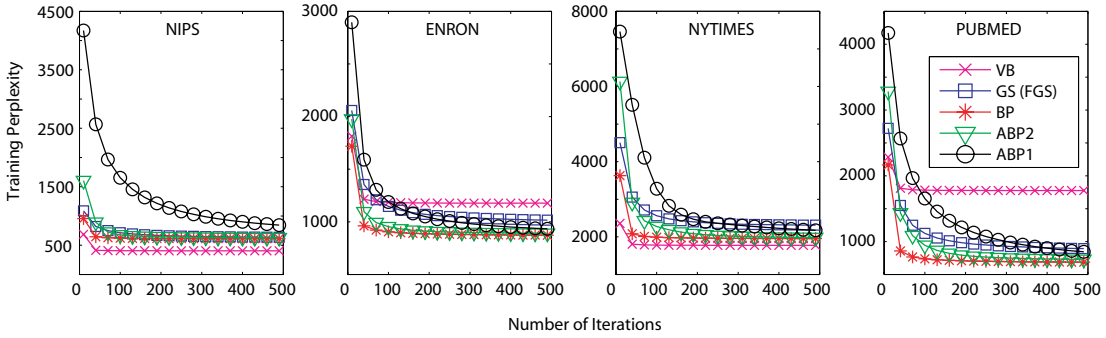

Fig. 9. Training perplexity as a function of the number of iterations when $K = 500$ on NIPS, ENRON, NYTIMES and PUBMED data sets.

ABP accelerates the topic modeling process significantly, and in the meanwhile remains a comparable topic modeling accuracy. As with the conventional BP algorithm, ABP scales linearly with the number of documents D and the number of topics K but with a fraction $\lambda_d \lambda_k \ll 1$.

In the various test sets, we find that at each iteration only 20% documents and 20% topics are enough to yield almost the same topic modeling performance as BP, which seems to follow the Pareto principle or 80-20 rule[1], stating that roughly 80% of the effects come from 20% of the causes. Furthermore, we find that ABP can choose even much smaller parameters λ_d and λ_k under the large D and K. This result implies that not all documents and topics contribute equally for the topic modeling performance of massive corpora with large number of topics. For example, at each iteration only 10% documents and 10% topics can provide comparable or much better performance than GS/FGS algorithms when $K = 900$. So, we speculate that the subsets $\lambda_d D$ and $\lambda_k K$ can be constants for massive data sets ($D \geq 10^6$) with large number of topics ($K \geq 1000$). In this way, ABP may use a fixed training time for probabilistic topic modeling of these massive data sets.

1. http://en.wikipedia.org/wiki/Pareto_principle

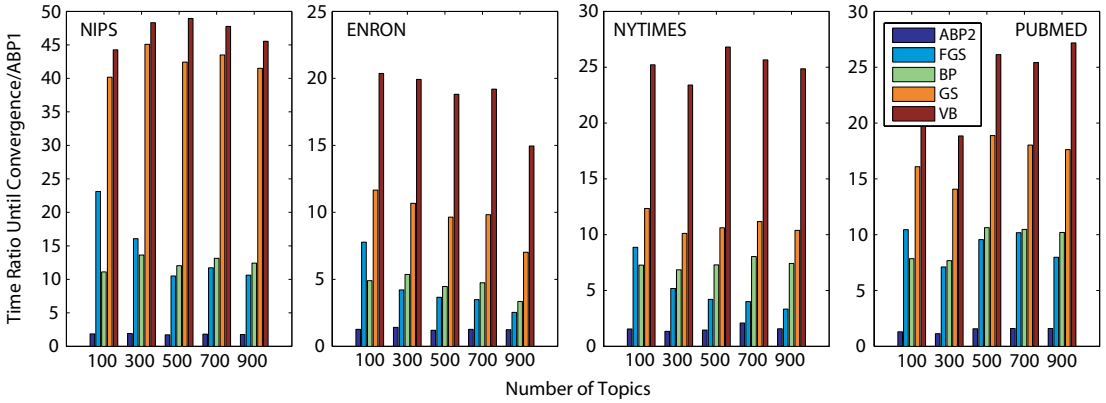

Fig. 10. Ratio of training time until convergence over ABP1 as a function of topics $K = \{100, 300, 500, 700, 900\}$ on NIPS, ENRON, NYTIMES and PUBMED data sets.

Topic 1	network recognition system training neural word set speech input classifier network recognition word system speech training set neural classifier model classifier recognition word set training classification speech model system class recognition training classifier set network word data classification speech system
Topic 2	circuit neuron chip signal input network output system analog neural signal circuit system chip neural analog output network input current circuit neural chip analog system input network output current voltage signal circuit chip analog system output neural input current sound
Topic 3	network function learning neural weight algorithm input error result linear network function learning weight neural input output unit error equation network unit input weight output training neural learning layer hidden network unit input weight output learning neural training layer hidden
Topic 4	neuron model input cell network spike synaptic firing system pattern neuron model cell input spike synaptic activity firing response pattern neuron model input cell synaptic spike network activity firing system neuron model cell input spike synaptic activity network firing response
Topic 5	learning function algorithm action policy problem control optimal step reinforcement learning algorithm function action problem result policy step theorem states learning control action system model task policy reinforcement step dynamic model control system learning movement controller robot motor position task
Topic 6	network unit learning weight input output training hidden set neural network unit learning input training weight output hidden task set function algorithm learning point problem linear number order result case function algorithm learning problem result action number bound policy set
Topic 7	model data parameter function algorithm distribution gaussian set network vector model data distribution parameter gaussian algorithm probability component method density model data distribution probability parameter gaussian method mean noise density model data distribution parameter method gaussian function set algorithm probability
Topic 8	image visual model field motion cell direction object map images image visual field object images motion map direction feature cell visual model field motion cell image direction object map eye image visual field motion cell images object map model direction
Topic 9	model control system network learning neural movement input motor position model control system learning movement robot controller motor position eye signal frequency component filter sound system auditory information analysis data set model algorithm structure problem representation rules graph vector cluster
Topic 10	data model set error training algorithm network function method learning data set training error function algorithm vector method problem network vector data feature set image problem features point representation algorithm function learning point algorithm network vector linear matrix equation order

Fig. 11. Top ten words of ten topics on NIPS training set: VB (red), GS/FGS (blue), BP (black) and ABP (green).

Because of its ease of use and fast speed, ABP is a strong candidate for becoming the standard LDA training algorithm. Future work includes the relaxation of memory requirements for massive corpora. We may extend ABP within the block optimization framework [20], reading data as blocks from hard disk into memory. With fast speed and little memory, we may realize practical topic modeling softwares for massive personal data including documents, photos, videos on a common PC in the near future.

ACKNOWLEDGEMENTS

This work is substantially supported by NSFC (Grant No. 61003154), the Shanghai Key Laboratory of Intelligent Information Processing, China (Grant No. IIPL-2010-009), and a grant from Baidu to JZ, and a GRF grant from RGC UGC Hong Kong (GRF Project No.9041574) and a grant from City University of Hong Kong (Project No. 7008026) to ZQL.

REFERENCES

[1] D. M. Blei, "Introduction to probabilistic topic models," *Communications of the ACM*.
[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
[3] J. Zeng, W. K. Cheung, and J. Liu, "Learning topic models by belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, p. arXiv:1109.3437v3 [cs.LG], 2011.
[4] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
[5] A. Asuncion, M. Welling, P. Smyth, and Y. W. Teh, "On smoothing and inference for topic models," in *UAI*, 2009, pp. 27–34.
[6] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Natl. Acad. Sci.*, vol. 101, pp. 5228–5235, 2004.
[7] J. Winn and C. M. Bishop, "Variational message passing," *J. Mach. Learn. Res.*, vol. 6, pp. 661–694, 2005.
[8] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Inform. Theory*, vol. 47, no. 2, pp. 498–519, 2001.
[9] A. Asuncion, "Approximate Mean Field for Dirichlet-Based Models," in *ICML Workshop on Topic Models*, 2010.
[10] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: Informed scheduling for asynchronous message passing," in *UAI*, 2006, pp. 165–173.
[11] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.
[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977.
[13] K. R. Canini, L. Shi, and T. L. Griffths, "Online inference of topics with latent Dirichlet allocation," in *AISTATS*, 2009, pp. 65–72.
[14] M. Hoffman, D. Blei, and F. Bach, "Online learning for latent Dirichlet allocation," in *NIPS*, 2010, pp. 856–864.
[15] D. Newman, A. Asuncion, P. Smyth, and M. Welling, "Distributed algorithms for topic models," *J. Mach. Learn. Res.*, vol. 10, pp. 1801–1828, 2009.
[16] K. Zhai, J. Boyd-Graber, and N. Asadi, "Using variational inference and MapReduce to scale topic modeling," *arXiv:1107.3765v1 [cs.AI]*, 2011.
[17] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed Gibbs sampling for latent Dirichlet allocation," in *KDD*, 2008, pp. 569–577.
[18] G. Heinrich, "Parameter estimation for text analysis," University of Leipzig, Tech. Rep., 2008.
[19] J. Zeng, "TMBP: A topic modeling toolbox using belief propagation," *J. Mach. Learn. Res.*, p. arXiv:1201.0838v1 [cs.LG], 2012.
[20] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin, "Large linear classification when data cannot fit in memory," in *KDD*, 2010, pp. 833–842.
[21] J. Chang, J. Boyd-Graber, S. Gerris, C. Wang, and D. Blei, "Reading tea leaves: How humans interpret topic models," in *NIPS*, 2009, pp. 288–296.